組込み OSS 移植工程に対する MCMC およびハザードレートモデルに基づく信頼性評価法 (不確実・不確定環境下における数理的意思決定とその周辺)

AUTHOR(S):

田村, 慶信; 山田, 茂

数理解析研究所講究録
第1802巻 2012年 120-126

組込みOSS移植工程に対するMCMCおよびハザードレートモデルに基づく信頼性評価法

山口大学大学院・理工学研究科 田村 慶信 (Yoshinobu Tamura) †
†Graduate School of Science and Engineering, Yamaguchi University
鳥取大学大学院・工学研究科 山田 茂 (Shigeru Yamada) ‡
‡Graduate School of Engineering, Tottori University

1 はじめに

オープンソースソフトウェア (open source software, 以下OSSと略す) はサーバ用途だけではなく，アプリケーションソフトウェア，組込みソフトウェアなど，多くの分野において採用されている．特に，OSSは，世界中の誰もが開発に参加でき，ソースコードが公開され，誰でも自由に改変可能なソフトウェアであることから，急激に普及が広まっている [1–3]．最近のOSSの傾向として，組込み機器に対してもAndroid [4] やBusyBox [5] に代表される組込みOSSが積極的に採用されつつある．

一方，OSSの利用に関しては，未だに多くの不安が残されている．まず第1に，システム導入後のサポートや品質上の問題といった利用者側の一般的な不安である．第2に，OSSは本当にビジネスになるのか，オープンソースのソフトウェアを事業化することによって自社製のソフトウェア商品までが市場を失うことにならないか，といった開発者側の不安である [6]．特にサポートや品質上の問題については，OSSの普及を妨げる大きな要因として考えられており，組込みOSSに関しては不安材料の大きな要因の1つとして考えられる．また，OSSは常にバージョンアップが繰り返されており，複数のバージョンがプロジェクトのWeb上に公開されている．この中から最適なバージョンを選択することが難しいことも問題点の一つとして挙げられる．

これまでのOSSに対する現在の研究動向としては，設計工程や開発手法，セキュリティを対象とした文献はいくつか提案されているが [7–10]，動的解析に基づいた組込みOSSに対する信頼性評価に関する研究はほとんど行われていないのが現状である．さらに，OSSの信頼性評価に関する特徴として，サーバおよびアプリケーションソフトウェアについては信頼度成長曲線に関して一定の傾向を示すものが多いが [11,12]，組込みソフトウェアについては，ハードウェアに依存するコンポーネントが含まれていることから，信頼性を評価することが難しくなってくる．従来から，ソフトウェア製品の開発プロセスにおけるテスト進捗管理や出荷品質の把握のための信頼性評価を行うアプローチとして，ソフトウェア故障の発生現象を不確定事象として捉えて確率・統計論的に取り扱う方法がとられている．その代表的かつ古典的モデルの1つとして，ハザードレートモデルがある [13–19]．

本論文では，こうしたオープンソースプロジェクトの下で開発されている組込みOSSを採用する際の移植作業工程 (ポーティング) に対する信頼性および移植性評価法を提案する．特に，実際に公開されている組込みOSSを採用した移植作業工程における信頼性評価法の適用可能性について考察する．これにより，組込みOSSの普及を妨げる大きな要因として考えられている品質上の問題に対して，信頼性という観点から定量的指標を提示することが可能となるものと考える．

2 組込みOSSの移植工程に対する要求仕様の変更を考慮したハザードレートモデル

本論文では，組込みOSSのポーティング時における動的実行環境，すなわち独自に開発されたハードウェアに対する組込みOSSの移植作業中に生じるソフトウェア故障には，次の2種類があるものと仮定する．

A1. 組込みOSSに潜在するフォールトにより引き起こされるソフトウェア故障．

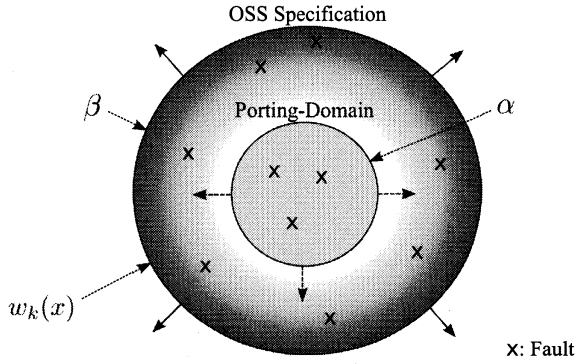

図 1： OSS の要求仕様と移植領域に関する概念図

A2. 独自に開発されたソフトウェアコンポーネントに内在するフォールトにより引き起こされるソフトウェア故障.

ここで，1 つのソフトウェア故障は 1 個のフォールトにより引き起こされると仮定し，発生したソフトウェア故障の原因となるフォールトは A1 と A2 のどちらによるものか区別できないものとする.

ここで，A1 のソフトウェア故障は確率 p_0 で発生し，A2 のソフトウェア故障は確率 p_1 で発生するものとする. このとき，$(k-1)$ 番目と k 番目のソフトウェア故障の時間間隔を確率変数 X_k とすると，X_k に対するハザードレート関数 $z_k(x)$ を，

$$z_k(x) = p_0\phi_0\{w_k(x)-(k-1)\}+p_1\phi_1\{N-(k-1)\} \quad (1)$$
$$(k=1,\ 2,\ \cdots,\ \alpha;\ \phi_0>0,\ \phi_1>0,\ N>0,\ p_0+p_1=1),$$
$$w_k(x) = \alpha e^{\beta k} \qquad (0<\alpha<1,\ -1<\beta<1), \quad (2)$$

により表すことができるものと仮定する. ここで，各諸量を次のように定義する.

$z_k(x)$ ：組込みソフトウェア全体に対するハザードレート,

p_0 ：組込み OSS に対する開発労力の割合,

p_1 ：A2 に対する開発労力の割合,

ϕ_0 ：組込み OSS に対する固有フォールト 1 個当りのハザードレート,

ϕ_1 ：A2 に対する固有フォールト 1 個当りのハザードレート,

$w_k(x)$ ：OSS の要求仕様と移植作業領域を考慮した組込みソフトウェア内に潜在する総固有フォールト数,

α ：組込み OSS 内に潜在する総固有フォールト数,

β ：OSS の要求仕様の変化率.

式 (1) は，組込みシステムに適用される OSS の要求仕様の変化を考慮した組込みソフトウェア全体に対するハザードレートを表す. さらに，本モデルに含まれる式 (2) は，図 1 に示すように，OSS の要求仕様と移植作業領域の変化を同時に考慮した組込みソフトウェア全体に対するフォールト数を表し，OSS の要求仕様の変化率に伴い増加または減少するものと定義する [20].

3 信頼性評価尺度

ポーティングの際の動的実行環境において，$(k-1)$ 番目と k 番目の間のソフトウェア故障発生時間間隔を表す $X_k(k=1,\ 2,\ \cdots)$ の分布関数は，

$$F_k(x) \equiv \Pr\{X_k \le x\} \quad (x \ge 0), \quad (3)$$

により定義され，時間区間 $(0,\ x]$ でソフトウェア故障の発生する確率を表す．ここで，$\Pr\{\mathrm{A}\}$ は事象 A の生起確率を表す．したがって，$F_k(x)$ の導関数

$$f_k(x) \equiv \frac{dF_k(x)}{dx}, \tag{4}$$

は，X_k の確率密度関数である．また，時間区間 $(0,\ x]$ でソフトウェア故障の発生しない確率を表すソフトウェア信頼度は，

$$R_k(x) \equiv \Pr\{X_k > x\} = 1 - F_k(x), \tag{5}$$

により定義される．式 (3) および式 (4) から，時間区間 $(0,\ x]$ でソフトウェア故障が発生していないときに，引き続く単位時間内にソフトウェア故障が発生する割合を意味するソフトウェア故障率（ハザードレート）を

$$z_k(x) \equiv \frac{f_k(x)}{1 - F_k(x)} = \frac{f_k(x)}{R_k(x)}, \tag{6}$$

により与えることができる [15].

したがって，式 (1) のハザードレートモデルから，信頼性評価尺度を導出することができる．確率密度関数は，

$$\begin{aligned} f_k(x) &= \{p_0\phi_0\{w_k(x) - (k-1)\} + p_1\phi_1\{N - (k-1)\}\} \\ &\cdot \exp\Big[- \{p_0\phi_0\{w_k(x) - (k-1)\} + p_1\phi_1\{N - (k-1)\}\} \cdot x \Big], \end{aligned} \tag{7}$$

となる．また，ソフトウェア信頼度は，

$$R_k(x) = \exp\Big[- \{p_0\phi_0\{w_k(x) - (k-1)\} + p_1\phi_1\{N - (k-1)\}\} \cdot x \Big], \tag{8}$$

と表すことができる．さらに，式 (7) から，X_k の平均値すなわち k 番目のソフトウェア故障に対する平均ソフトウェア故障時間間隔（Mean Time between Software Failures，以下 MTBF を略す）は，

$$\mathrm{E}[X_k] = \frac{1}{p_0\phi_0\{w_k(x) - (k-1)\} + p_1\phi_1\{N - (k-1)\}}, \tag{9}$$

により与えられる．

OSS の要求仕様の変化に伴う潜在フォールト数を評価することは，組込みソフトウェアの移植工程を管理する際において重要となる．OSS の要求仕様の変化を考慮した組込みソフトウェア内における潜在フォールト数は，

$$w_k(x) = \alpha e^{\beta k} + N, \tag{10}$$

となる．また，OSS の要求仕様の変化を考慮した組込みソフトウェア内における残存フォールト数は，

$$N(k) = \alpha e^{\beta k} + N - k, \tag{11}$$

により与えられる．

4 OSS の各コンポーネントに対するベイズ定理の適用

本論文では，OSS のコンポーネント別フォールト発見比率に対してベイズの定理を応用する．まず，ある重要度に対する時刻 t におけるフォールト発見比率を y_t とし，y_t から θ_t を予測する場合を考える．この場合，事前分布として時刻 $(t-1)$ までの結果を使用する．例えば，データ D とは独立に，データを取得する前の事前情報として θ に関する何らかの知見がある場合，データを得て更新された情報は，ベイズの定理により，

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)d\theta} \propto p(D|\theta)p(\theta), \tag{12}$$

となることが知られている．

本論文では，逐次データ y_t に対する θ_t の予測において，過去のフォールト発見比率に関するデータから，現時点における θ_t を推定することを考えた場合，式 (12) から，以下のように表現できる．

$$p(\theta_t|y_1,y_2,\cdots,y_t)=\frac{p(y_t|\theta_t)p(\theta_t|y_1,y_2,\cdots,y_{t-1})}{\int p(y_t|\theta_t)p(\theta_t|y_1,y_2,\cdots,y_{t-1})d\theta_t}. \tag{13}$$

ここで，式 (13) に対して，$p(\theta_{t-1}|y_1,y_2,\cdots,y_{t-1})$ から $p(\theta_t|y_1,y_2,\cdots,y_t)$ のように逐次的に更新する場合を考慮し，

$$p(\theta_t|y_1,y_2,\cdots,y_t)=\frac{p(y_t|\theta_t)\int p(y_t|\theta_{t-1})p(\theta_{t-1}|y_1,y_2,\cdots,y_{t-1})d\theta_{t-1}}{\int p(y_t|\theta_t)p(\theta_t|y_1,y_2,\cdots,y_{t-1})d\theta_t}, \tag{14}$$

と定義する．これは，時刻 $t-1$ におけるフォールト発見比率 θ_{t-1} が，時刻 t において，どのような確率となり得るかを表現している．本論文では，

$$\theta_t=\theta_{t-1}+\sigma, \tag{15}$$

のように単純な場合を仮定する．ここで，σ は独立なガウス型雑音を表す．

5 各コンポーネントに対するフォールト発見比率の推定

MCMC（Markov chain Monte Carlo methods，以下 MCMC と略す）は，乱数を発生させ，マルコフ連鎖を使用して確率分布のサンプリングを行う手法の 1 つである．基本的に，多変量分布からの確率変数のサンプリングは困難であることが多いが，MCMC では，比較的容易に目標分布からの確率標本を発生することができる．本論文では，Metropolis-Hastings（MH）アルゴリズムを適用する．MH アルゴリズムの手順を以下に示す．

1. $\theta_t=\theta$ のとき，採用された密度 $q(\theta,\theta')$ により θ' を生成する．
2. θ' を $\alpha(\theta,\theta')>u$ で θ_{t+1} として置き換える．ただし，$\alpha(\theta,\theta')\leq u$ の場合は $\theta_{t+1}=\theta$ とする．
3. 初期値の影響がある部分がなくなるまで上述のプロセルを十分長い間繰り返す．

MCMC を適用するにあたり，本論文では簡単のために各フォールト重要度に関するフォールト発見比率が以下の確率密度関数をもつ正規分布に従うものと仮定する．

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{16}$$

ここで，μ は平均，σ^2 は分散を表す．μ および σ により構成される事前分布・超事前分布という階層により，パラメータ間の関係を考慮した階層ベイズモデルを考える．

6 数値例

6.1 組込み OSS

本論文では，携帯電話用 OS として開発・公開されている Android [4] 上で BusyBox [5] が動作するシステムを構築する環境を想定する．移植作業工程を想定するために，実際の Android および BusyBox のオープンソースプロジェクトにおけるバグトラッキングシステム上に登録されたフォールトデータを適用した数値例を示す．Android は携帯電話用 OS として知られ，BusyBox はテレビ，オーディオ，ブロードバンドルータ，小型サーバなど，家電製品を代表とした様々な組込み製品に利用されている．

6.2 信頼性評価

移植工程に影響を与える主要コンポーネントとして Android，Busybox，および buildroot を取り上げる．各コンポーネントに対する MCMC によるフォールト発見比率に関する推定結果を，図 2～図 4 に示す．これらの結果から，Android に関するフォールト出現率が最も大きく，Busybox のフォールト出現頻度が最も小さい様子が確

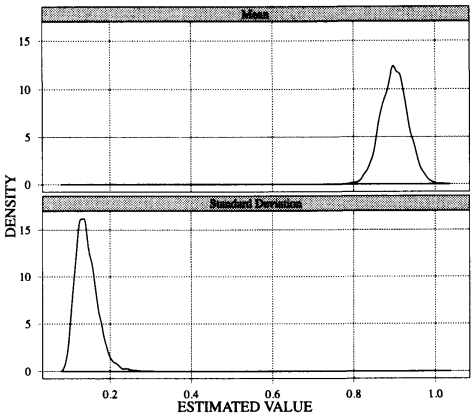

図 2： “Android” コンポーネントに対する推定結果.

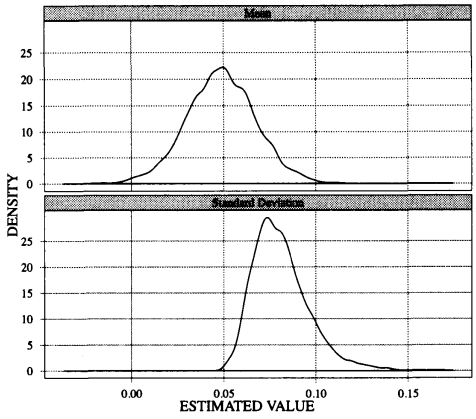

図 3： “Busybox” コンポーネントに対する推定結果.

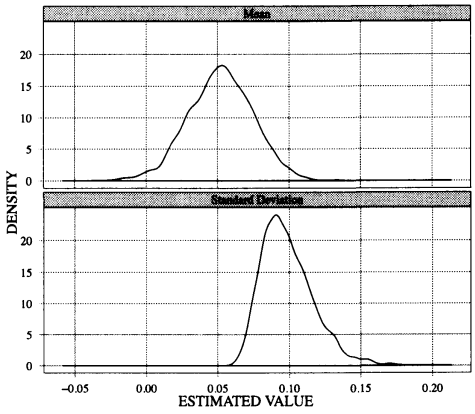

図 4： “buildroot” コンポーネントに対する推定結果.

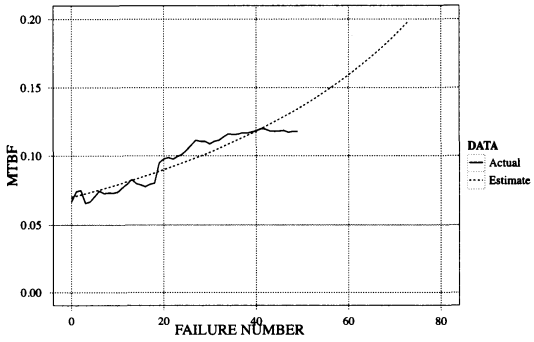

図 5： 推定された MTBF.

認できる．また，Android に関する標準偏差は大きく，フォールト出現頻度についてもばらつきがあることが確認できる．このことから，移植工程において Android を重点的に管理し，Busybox よりも buildroot に注意しつつ移植作業を行う必要があることが分かる．

次に，推定された MTBF を図 5 に示す．図 5 から，フォールトが発見されるにつれて MTBF の値が増加していく，すなわち信頼度成長が起こっている様子が確認できる．逆に，MTBF が時間の経過とともに減少し信頼度が退化する場合は，移植作業が失敗に終わる可能性が高いため，別バージョンの組込み OSS を導入するなどの検討を行う必要がある．

さらに，要求仕様の変化を考慮した場合における推定された潜在および残存フォールト数を図 6 に示す．図 6 から，β が負の値の場合は残存フォールト数が減少する様子が確認できる．一方，本モデルにおいては，β が正の値をとる場合は残存フォールト数が増加する．このことから，本モデルにより信頼度成長と退化の両方を包括することが可能となり，OSS の移植工程において，予め適切な組込み OSS のバージョンを選択する際の評価尺度として利用できるものと考える．

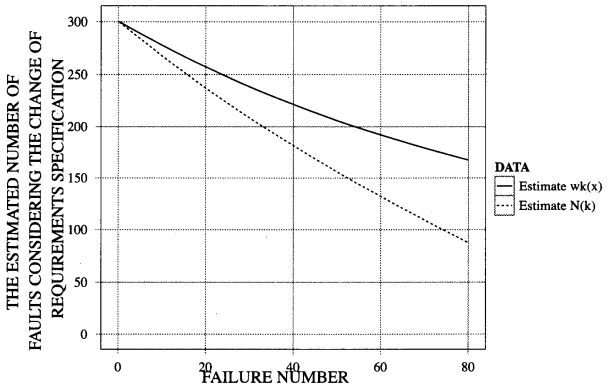

図 6： 要求仕様の変化を考慮した場合における推定された潜在および残存フォールト数.

7 むすび

本論文では，オープンソースプロジェクトの下で開発されている組込み OSS の移植作業工程に対する要求仕様の変化を考慮したハザードレートモデルに基づく信頼性評価ツールを開発した．また，実際の OSS のバグトラッキングシステムに登録されているフォールトデータに対して，信頼性評価尺度に関する数値例を示した．組込み OSS を利用した組込みシステム開発においては，移植作業が成功するか否かが，組込み製品が出荷できるかどうかに直接的に関係してくることから，組込みシステムの開発工程の中でも移植工程を適切に管理することは非常に重要となる．特に，組込み OSS の故障発生時間間隔データに関しては，多くのフォールトが発見されるにつれて MTBF が増加するという傾向があるものとそうでないものとが存在するため，それに応じた適切なハザードレートモデルを選択する必要がある．本論文では，組込み OSS とデバイスドライバのようなコンポーネントの 2 種類を想定したハザードレートモデルを提案した．

組込み OSS の普及の流れを阻害する要因として，サポートや品質上の問題が挙げられる．本論文では，このような問題を解決するためにオープンソースプロジェクトの下で開発された組込み OSS の移植作業工程に対する信頼性評価法の 1 例を示した．本論文の数値例で取り上げた Android および BusyBox は，機器のネットワーク化，開発コスト削減，オープンソースといった点から組込み OS として近年注目されている．今後もオープンソースプロジェクトに基づく開発形態は急速に発展するものと考えられることから，こうした組込み OSS の信頼性および移植性評価法として利用できるものと考える．

謝辞

本研究の一部は，文部科学省科学研究費基盤研究 (C)（課題番号 22510150）の援助を受けたことを付記する．

参考文献

[1] The Apache HTTP Server Project, The Apache Software Foundation. [Online]. Available: http://httpd.apache.org/

[2] Oracle Corporation and/or Its Affiliates, MySQL. [Online]. Available: http://mysql.com/

[3] The OpenStack project, OpenStack. [Online]. Available: http://www.openstack.org/

[4] Open Handset Alliance, Android. [Online]. Available: http://www.android.com/

[5] Erik Andersen, BUSYBOX. [Online]. Available: http://www.busybox.net/

[6] ソフトウェア情報センター研究会報告書, オープンソースソフトウエアの利用状況調査／導入検討ガイドラインの公表について, 東京, 2004.

[7] A. MacCormack, J. Rusnak, and C.Y. Baldwin, "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code," *Informs J. Management Science*, vol. 52, no. 7, pp. 1015–1030, 2006.

[8] Y. Zhoum, J. Davis, "Open source software reliability model: an empirical approach," *Proceedings of the Workshop on Open Source Software Engineering (WOSSE)*, vol. 30, no. 4, 2005, pp. 67–72.

[9] P. Li, M. Shaw, J. Herbsleb, B. Ray, and P. Santhanam, "Empirical evaluation of defect projection models for widely-deployed production software systems," *Proceeding of the 12th International Symposium on the Foundations of Software Engineering (FSE-12)*, 2004, pp. 263–272.

[10] J. Norris, "Mission-critical development with open source software," *IEEE Software Magazine*, vol. 21, no. 1, 2004, pp. 42–49.

[11] Y. Tamura and S. Yamada, "Software reliability assessment and optimal version-upgrade problem for open source software," Proc. the 2007 IEEE International Conference on Systems, Man, and Cybernetics, Montreal, Canada, October 7–10, 2007, pp. 1333–1338.

[12] Y. Tamura and S. Yamada, "A method of user-oriented reliability assessment for open source software and its applications," Proc. the 2006 IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan, Oct. 8–11, 2006, pp. 2185–2190.

[13] M.R. Lyu, ed., *Handbook of Software Reliability Engineering*, IEEE Computer Society Press, Los Alamitos, CA, 1996.

[14] J.D. Musa, A. Iannino, and K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York, 1987.

[15] 山田 茂, ソフトウェア信頼性の基礎ーモデリングアプローチー, 共立出版, 東京, 2011.

[16] G.J. Schick and R.W. Wolverton, "An Analysis of Competing Software Reliability Models," *IEEE Transactions Software Engineering*, vol. SE-4, Issue 2, pp. 104–120, 1978.

[17] Z. Jelinski and P.B. Moranda, *Software Reliability Research, in Statistical Computer Performance Evaluation*, Freiberger, pp. 465–484, Academic Press, New York, 1972.

[18] P. B. Moranda, "Event–altered Rate Models for General Reliability Analysis," *IEEE Transactions Reliability*, vol. R–28, no 5, pp. 376–381, 1979.

[19] M. Xie, "On a Generalization of the J-M Model", *Proceedings Reliability '89*, 5 Ba/3/1–5 Ba/3/7, 1989.

[20] S. Yamada and T. Fujiwara, "Testing-domain dependent software reliability growth models and their comparisons of goodness-of-fit," *International Journal of Reliability, Quality and Safety Engineering*, vol. 8, no. 3, pp. 205–218, 2001.